Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Prototype of an augmented reality user manual app

by

Filip Källström & Fredrik Palm

LIU-IDA/LITH-EX-A--14/039--SE

2014-06-26

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Final thesis

Prototype of an augmented reality user manual app

by

Filip Källström & Fredrik Palm

LIU-IDA/LITH-EX-A–14/039–SE

2014-06-26

Supervisor: Erik Berglund

Examiner: Anders Fröberg

Abstract

This thesis describes how augmented reality can be used when developing an instructional application. After studying augmented reality apps and papers, a prototype for mobile devices was developed to discover the possibilities that augmented reality offers and show how issues inherent to the technology can be solved. The app was developed with usability in mind, with consideration for how well suited each feature was for augmented reality.

Our results show that it is possible to use augmented reality in instructional apps, but that there are issues to consider when working with augmented reality. Among others, we discuss how to deal with the three dimensionality of the interface, augmented reality's physical requirements, and the quality of the tracking that aligns the interface with the real world. Augmented reality also enables plenty of new functionality for apps, like the ability to use physical movement as input and to essentially bind information to a real, physical place.

The app was tested and built for an advanced machine. We built guides that use animated instructions to teach the user how to complete a task. There is also an information view that displays details about parts of the machine and an overview that helps the user find parts. We also took efforts to generalize the process, so that the app can be adjusted to suit a variety of products.

Contents

1 Acknowledgments **1**

2 Introduction **2**
2.1 Problem Description 2
2.2 Purpose 3
2.3 Scope and Limitations 3

3 Glossary **4**

4 Theory **5**
4.1 Introduction to Augmented Reality 5
4.2 Tracking 6
4.2.1 Sensor-based Tracking 6
4.2.2 Vision-based Tracking 6
4.2.3 Detectability of Natural Feature Targets 7
4.2.4 Extendible Tracking 9
4.3 Interaction Techniques 9
4.3.1 Embodied interaction 9
4.3.2 Tangible Interaction 10
4.4 Challenges with AR 10
4.4.1 The Effect of Camera, Lighting and Quality of Target 10
4.4.2 Placement and Size of Targets 11
4.4.3 Scaling and Interface Legibility 11
4.4.4 Physical Demands 12
4.5 Similar Products 12
4.5.1 Anatomy 4D by Daqri 13
4.5.2 Audi eKurzinfo 14

5 Method **16**
5.1 Literature Study 16
5.2 Brainstorming 17
5.3 Design and Development 17

5.4 Development Tools 18
5.4.1 Vuforia 18
5.4.2 Unity 4 19
5.4.3 Graphics 19

6 Result **20**
6.1 Detection and Tracking 20
6.1.1 Camera Management 20
6.1.2 Target Selection 20
6.1.3 Target Management 23
6.2 Using References 24
6.3 Generalization 25
6.3.1 Coordinate and Information Databases 25
6.3.2 Database Implementation 26
6.4 Interface 27
6.4.1 Robustness 27
6.4.2 Scaling and Interface Legibility 27
6.4.3 Text Legibility 31
6.4.4 User Tutorial 32
6.4.5 GUI 34
6.5 Interaction 36
6.5.1 Embodied Touch Interaction 36
6.5.2 Tangible Interaction with Virtual Buttons . . . 36
6.5.3 Ability to Freeze the Display 37
6.6 Machine Overview 38
6.6.1 Use of the Machine Overview in AR 40
6.6.2 Implementation of Machine Overview in AR . . 41
6.7 Information Viewer 41
6.7.1 Implementation 42
6.8 Guide Systems 42
6.8.1 AR Guides 42
6.8.2 GUI Guides 45
6.8.3 Implementation of the Guides 46

7 Discussion **47**
7.1 Method 47
7.1.1 Iterative approach 47
7.1.2 Brainstorming 48
7.1.3 Kanban 48
7.1.4 Use of Vuforia and Unity 48

7.2 Result 49
7.2.1 Quality of Tracking 49
7.2.2 Measuring the Quality of Targets 50
7.2.3 Use of Augmented Reality 50
7.2.4 Detail Layers 51
7.2.5 AR Apps as User Manuals 52

8 Conclusions **54**
8.1 Using Augmented Reality 54
8.2 Improving Usability 54
8.3 Future Work 55
8.3.1 Other Tracking Techniques 55
8.3.2 Continuation of Generalization 55
8.3.3 Development for Wearable Devices 56

9 References **57**

1 Acknowledgments

We would like to thank our supervisor, Erik Berglund, and our examiner, Anders Fröberg, for their support and advice during the project.

We would also like to thank everyone at Attentec, especially our supervisors Kajsa Goffrich and Gustav Hjortsparre, for welcoming us to the company and helping out with their expertize and resources, and VSM Group for their interest in the project, and help with advice and resources.

Lastly, we would like to thank Niclas Olofsson and Peter Larsson-Green for their help reviewing this report.

2 Introduction

The massively increased power and capability of commonly available and used mobile devices during the last decade gives developers the opportunity to use augmented reality (AR) to enhance the functionality of mobile applications. Augmented reality has many advantages and presents the unique possibility of annotating the real world with digital elements, giving users easy access to information in an intuitive way.

AR can be used in many different areas. The focus in this thesis will be on AR for instructional use. Specifically we are looking at using AR in digital user manuals. That means that the digital data that is superimposed on the real objects are meant as guidance for a user on how to perform a task.

This thesis was conducted at Attentec, a software development consulting firm, together with VSM Group AB, a manufacturer of advanced, top-of-the-line sewing machines. Both parties were interested in the possibility of using augmented reality in some form to increase the value of a product, and this thesis intends to demonstrate how that could be done.

2.1 Problem Description

In this thesis we are looking at ways to take advantage of augmented reality when producing a digital user manual. By using a mobile application with augmented reality it is conceivable to provide a digital, interactive user manual that makes the process of using the manual more enjoyable and immediate by superimposing the relevant information and instructions visually, directly on the product.

However, many challenges arise when using augmented reality. The way to best design and use AR applications is new, both to developers and users and effort is required to understand what is possible in the medium. There are problems with how to build the interface since it changes depending on where the user is targeting the camera, and with teaching the user how to position the camera in order for detection

and tracking to work well. These problems and more will be discussed in the thesis. We propose solutions and present them in a prototype application for Android.

2.2 Purpose

The purpose of this thesis is to design and develop a prototype of an instructional application using augmented reality, presenting and proposing solutions to many of the difficulties that are introduced when using augmented reality in mobile applications. The questions studied in this thesis are:

- How can a mobile application use augmented reality to show how a certain tool is used?
- What can be done to improve the usability of instructional augmented reality applications?

2.3 Scope and Limitations

Augmented reality is a broad area with several different hardware and software solutions. In terms of hardware, this thesis is first and foremost focused on mobile AR, specifically augmented reality for Android, and does not take head-worn devices such as glasses, or other such hardware into account when discussing problems with AR and their solutions.

We use existing frameworks to handle detection and tracking. Computer vision is not part of this thesis's scope, although we do discuss it where it has an effect on the design of the AR app.

The application built during this project is a prototype, and not intended to be used or evaluated as a finished product.

3 Glossary

AR Layer The layer of data and interface elements that is superimposed on the target when tracking.

Augmented Reality (AR) A view of the real world, augmented with virtual elements.

Marker Target A type of target that is specifically designed for image detection, e.g QR-codes.

Natural Feature Target A type of target that is not necessarily designed for augmented reality, e.g photo, logos.

Pose The transform, i.e the position, size and rotation, of an object in relation to the camera.

(Image) Target An object that the application can recognize, detect, trigger on and track.

Tracking When the app has triggered on a target, the process of continuously calculating the target's pose.

Triggering When the app detects and begins to track a target.

4 Theory

This chapter contains the background theory required to understand the content of this thesis. We also discuss similar products that have inspired our prototype.

4.1 Introduction to Augmented Reality

Augmented reality is about enhancing the real world with virtual data. Applications recognizes the surroundings of a user and increases its value by adding different kinds of virtual data. The most common applications of AR uses it to add visual graphics and text to the user's surroundings. However, AR is not limited to just visual media. Any virtual data such as audio or haptics can be used to enhance the world. Nor is it limited to any specific hardware. [5]

Below is an attempt by FitzGerald et al. to capture a broader definition of augmented reality:

> "... being able to augment one's immediate surroundings with electronic data or information, in a variety of media formats that include not only visual/graphic media but also text, audio, video and haptic overlays." [7]

Augmented reality is a young technology, especially in the subject of mobile augmented reality. In their 2012 paper, Gervautz and Schmalstieg's conclusions are that "only a fraction of the user interface design space afforded by 3D interaction with the environment through AR has been explored" and that "even with a flawless AR technical implementation, researchers will still be faced with the problem of insufficient knowledge about AR user interface design" [9]. The first major research on AR in general was done about 20 years ago. At this time most of the research was done towards head-worn AR and the actual hardware for the AR was mainly found in research laboratories. Since then the market has changed dramatically and today devices that can be used for AR already exist in many people's hands

in the form of smartphones. However, there are still interesting things happening with head-worn AR, especially with the announcement of Google Glass[1] and other augmented reality glasses. [3, 5]

4.2 Tracking

Tracking is used to connect the virtual and physical worlds. When the world moves, that movement should be tracked so that the virtual elements can keep their connection to it.

4.2.1 Sensor-based Tracking

Sensor-based tracking is commonly used in AR. This method uses sensors such as GPS, compass, gyroscope and accelerometers to estimate the position of physical objects. The sensor-based method is often used to give information about nearby places. The GPS is used to retrieve the location of the user in relation to the target and sensors are often used to calculate the orientation of the mobile device. Notes such as distance and general information about the targeted location can be placed on the display in the direction of the target. Examples include Wikitude[2], which shows information about nearby point of interest, and Google's Sky Map[3] which uses GPS and compass to display the names of the stars and constellations that the user is looking at through the device. [26, 5]

4.2.2 Vision-based Tracking

To be able to give information about the user's immediate surroundings, cameras can be used as input for AR. By analyzing a camera's output in real-time, it is possible to detect and track physical objects, *targets*, and use these to place virtual objects, *augmentations*, on a screen.

This is the type of AR explored in this thesis and what we will be referring to from now on when we are talking about tracking unless specified otherwise. Examples of apps using vision-based tracking will be discussed in section 4.5.

[1] http://www.google.com/glass/start/
[2] http://www.wikitude.com/app/
[3] http://www.google.com/mobile/skymap/

Markers are targets that are specifically designed to be detectable by image recognition. Markers are analyzed beforehand so that recognizable feature patterns are stored for the image recognition process to use. During runtime the image recognizer will then search for and match similar patterns in the camera images. A common example of a marker is the QR-code whose corners, the 'finder pattern', are designed to be detected easily, and are shared among all QR-codes [18]. [13]

The figures below, 4.1 and 4.2, show two different kinds of markers, both designed for quick detection.

Figure 4.1: A QR-code. QR-codes can also store embedded data.

Figure 4.2: A frame marker provided by the Vuforia AR SDK.

In some cases, it is not possible or desirable to place a marker on or near the object that is to be augmented. When that is the case, natural feature targets can be used instead. Natural features are elements of physical objects that can be made detectable, for instance by taking a photo of it, without the addition of a marker. [12, 13]

Since QR-codes and markers are designed for image recognition they can make the tracking more robust and require less computational effort in comparison to natural feature targets. However they suffer from having to be placed beforehand on the targeted machine. [26]

4.2.3 Detectability of Natural Feature Targets

Detection and tracking are essential for vision-based augmented reality apps. If the app fails to trigger, the augmented reality layer can not be placed. There are a number of factors that determine the detectability of targets, some of which are more difficult for designers to control.

According to three leading AR frameworks, natural feature targets should be chosen, or premeditatively designed, based on several important properties. High local contrast of the target is important because contrast increases the ability to discern certain detectable features of the target. The target should also have low reflectivity since reflective surfaces can change how the target looks and therefore introduce unwanted features depending on the target's current environment. It is also important for the target to have high levels of detail, preferably distributed over the whole target so that many detectable features exist and can be tracked. Finally the details in the target should have low symmetry, since high symmetry can make it hard or even impossible to map a specific detectable feature to the correct position on the target. [17, 14, 10]

One of the augmented reality API's available, Vuforia, has a target manager that will rate images uploaded by the developer to inform of how "augmentable" targets are and give suggestions on what to improve, as shown in figure 4.3 below.

attentec
Target ID: 0f96397ae6d248068be878ec1e16740a
Augmentable:
Increase the total number of features in the image by adding more visual detail to the whole scene
Ensure that all objects have clearly defined edges and high local contrast
Improve the overall feature distribution by adding textured objects in the empty spaces of the image
Added: Feb 17, 2014
Modified: Feb 17, 2014

Figure 4.3: Vuforia's rating of a photo of the Attentec logo. The image shows the original photo on the left side of the dashed line and features found by Vuforia as yellow plus signs on the right side.

In figure 4.3, you can see that detectable features, represented as yellow plus signs, are found at the edges of the letters. Round shapes, like the *c* in *Attentec* generate fewer features than shapes with hard edges, like the *t*.

With natural features you will have to consider not just how augmentable the object itself is, but also the photos of the object that are used to generate the target. The photos can be edited to increase local contrast which may help the detection. [16]

4.2.4 Extendible Tracking

Taking the idea of natural feature targets one step further, *extended* or *extendible* tracking captures and integrates new targets into the tracking database in run-time [23]. When tracking a predefined target, new unprepared targets can be created from its surroundings by calculating their 3D position based on the active target, potentially making tracking more stable and covering a larger area. This requires that the surroundings are static. Early experiments showed that extendible tracking was sensitive to propagation of tracking errors, due to the dynamic calibration of the 3D position [2].

Recently, extended tracking was added to Vuforia. The feature allows augmentations to persist even when tracking of a predefined target is lost, instead relying on dynamically added targets. [15]

4.3 Augmented Reality Interaction Techniques

Interaction with AR applications is not as straight forward as with normal applications. Smartphone interaction techniques such as touch, swipe and pinch-to-zoom are made more complicated as the AR interface exists in 3D and is connected to real world objects. It is not always easy to understand how an AR application acts when interacting with the screen or the targeted physical objects. We will explain two interaction techniques; *embodied interaction* and *tangible interaction*.

4.3.1 Embodied interaction

With embodied interaction the user performs actions using the mobile device to interact with the AR interface. Clicking on the display performs click actions on objects in the AR interface. By moving the mobile device in relation to the physical object the user can look around the object. Zoom is thus performed by simply moving the mobile device closer or further away from the object just as it works with a normal camera. Other possible embodied interactions are for example pinch-to-zoom and rotation actions performed on the display, however translating these to an AR interface can be complicated. For example, should a pinch-to-zoom gesture just zoom the AR interface or zoom the camera view as well? [9]

4.3.2 Tangible Interaction

Instead of interacting with the AR through the mobile device, tangible interaction have the user interact with targets in front of the mobile device. This could either be interaction with printed markers or interaction with actual objects detected via natural features. An example of tangible interaction is that the user uses his hand in front of the camera to pick up, rotate or in other ways change how the physical objects are positioned. At the same time the interface seen on the display will be moved or rotated as the user moves the real world objects. [9, 26]

Virtual button is a tangible interaction concept where a digital button shown on the mobile display can be pressed by "touching" the corresponding position of the button in the real world. The concept is that when the user occludes the part of a target that the button covers in the real world the application considers that button to be pressed. When the camera once again sees the features the button is considered released. Features can of course be undetected for other reasons than a user occluding it, causing unwanted click events of the virtual button. Also, occluding features will have a negative impact on the tracking of the target. [25]

4.4 Challenges with Augmented Reality

This section describes the challenges we have found that make it harder to develop good quality applications for mobile AR. We present solutions to these challenges in the result section.

4.4.1 The Effect of Camera, Lighting and Quality of Target

The success rate of the tracking system depends not just on the detectability of the targets, but also on the camera and the lighting in the room. The cameras available for mobile devices vary hugely and the quality of their output will directly affect how usable the app is. All cameras produce images that are distorted to some degree depending on the quality of the optics. Higher levels of distortion will make it more difficult to detect features. [13]

The lighting in the room is just as important. Without any light the app can be useless, and strong lights, such as the camera's flash,

will change the dynamic of the target. Other artificial lights can be beneficial to tracking quality [19].

4.4.2 Placement and Size of Targets

The layer that is superimposed on the target will be placed and oriented, or posed, to align it with the target. When the target moves, the layer should follow. Since the detection of the target is an error-prone process done repeatedly in real-time, the errors will cause the layer to move slightly even if the camera and target are stationary. A study in 2014 confirms that the deviation increases with the distance to the target [19]. This error will be more noticeable if the size of the AR layer is larger, meaning that the size and placement of the target will have a deciding effect on how the AR layer is designed, or vice versa. If the AR layer needs to cover a large area, it is best to have a large target placed in its center.

The imperfections of the posing from the tracking method results in a shaky and sometimes unreliable interface. The interface can jump and rotate unexpectedly, likely making it difficult to interact with. It is a good idea to design AR interfaces with these problems in mind. This is most true where the pose estimation error is largest.

4.4.3 Scaling and Interface Legibility

Unlike most, interfaces in augmented reality typically exist in 3D space. The user controls where it is viewed from by moving the camera, changing the angle and distance between the camera and the AR layer. This is part of what achieves the feeling that the interface exists in the real world, rather than just in the application. If the interface is not designed with this in mind, its elements will easily disappear behind the camera view or appear blurred when the camera is close. Due to this, the interface can easily appear cluttered and be illegible. Azuma and Furmanski described and evaluated methods for avoiding clutter by placing elements intelligently in real-time to limit overlap [1].

Text can also be illegible due to the variable background and light. There are several possible solutions to this problem, ranging from simple solutions like adding a shadow to more complex calculative solutions that optimally choose the color of the text. [8]

4.4.4 Physical Demands

Another difficulty with instructional mobile AR applications has to do with the user's physical limitations. Some AR applications require the user to hold the mobile camera and point it towards a machine. When adding the instructional aspect, the user is also expected to interact with the physical machine. In many cases this might not be a problem; the user can hold the camera with one hand and manage the machine with the other. However, there will also be many cases where the user needs to use both hands to execute the instructions on the machine. If the application is running on a tablet it may even be necessary for the user to use both hands to even be able to hold the tablet steady for a longer period of time. This leads to a conflict of what the user can do and what the application expects of the user. [22]

Another issue with interaction is the user's view of the machine. Ideally the user would be able to look at the augmented world on the mobile display and simultaneously be able to easily work with the machine. However, problems arise as the user might find it hard to perform actions on the actual machine while looking at the actions on a two dimensional display. Users may also find it hard to operate the machine as the camera has to be held in front of the machine in order to not lose the information shown with AR and may therefore be in the way of the user's vision and actions. [22]

A possible solution to these problems is called freeze. The idea is to let the user interact with the AR interface while the AR interface and the camera background is static. This avoids screen interface shaking as the user touches the display. It also allows the user to put the device down, freeing up both hands for operations on the machine. [4]

4.5 Similar Products

In this section we discuss several existing products in order to get an idea of what other developers are doing in this field. Unfortunately, augmented reality apps can be difficult to test because they often require some physical product to work. For instance, there are user manual apps for cars that we were unable to properly try on our own because doing so requires access to a car of a specific model.

4.5.1 Anatomy 4D by Daqri

Anatomy 4D is an educational augmented reality application for Android and iOS that shows a detailed 3D model of the human body, superimposed on a marker. According to the developer Daqri, the app has over 250,000 downloads. After a marker, which can be printed, is placed on a flat surface, the user can view the model of the body by looking at the marker through the camera, see figure 4.4 below. The user can control which parts of the anatomy are shown by pressing buttons in the GUI. [6]

Figure 4.4: A screenshot from a demonstration video by Daqri showing how their app works on an iPad.

The benefit of using AR in this case is debatable. We feel that the main benefit of the AR in this application is that it is easy to view the model from different angles by physically moving the camera around it. When tracking fails the app does not remove the model and instead transitions into a non-AR mode. In this mode, the model can be rotated, panned and zoomed by touch controls, like what has become the standard for both Android and iOS apps, with pinch and drag movements. Using these controls works fine, but it is noticeably slower than moving the physical camera. However, it is difficult to say that the improved camera movement outweighs the cost of using AR; e.g the requirement of printing a marker and dealing with tracking errors.

4.5.2 Audi eKurzinfo

Audi released an augmented reality user manual app developed by Metaio[4], an AR software developer, for their A1, A3 and S3 cars. The app recognizes individual parts of the car, such as different objects in the engine compartment, and displays information or maintenance instructions about the parts. The app uses vision-based tracking and can track both 2D images and 3D models. The idea is that the app should be able to reduce the need for big paper user manuals. [20]

According to the app's tutorial on how to use the app, the user first selects the appropriate car model. The app then instructs the user to "centre the desired item in the selection box". When the user moves the camera so that the car part of interest is in the selection box in the center of the camera view, the app should recognize it. When that occurs the name of the part is displayed on the screen. The box also turns red and the user can press a button to show information about that part, see the figure below. The information is then shown in a more traditional manner without AR.

Figure 4.5: Picture of the Audi AR app "eKurzinfo". The app has triggered on the cruise control system and shows information of it.

[4] http://www.metaio.com/

The main use of the app is to get information about any part of the car quickly, even if the name of it is unknown to the user. However in the description of the app the developers acknowledges that they cannot guarantee that the app recognizes the parts correctly and could display information about the wrong part [21]. This is of course a problem that may not be acceptable for apps designed to show service and maintenance information.

5 Method

Mobile AR interfaces are viewed and interacted with in three dimensional space and therefore difficult to prototype on paper. The user experience relies very much on the feeling of the interface. Because augmented reality is a relatively new technology, it is hard for developers and users to know what works and feels good before they have tried the functionalities in an actual AR application. That is why we have chosen to go for an iterative approach to developing a prototype application and testing our proposed solutions and design principles, starting with a literature study followed by iterating brainstorming sessions and development, see figure 5.1 below. This way we can get feedback on the design ideas quickly.

Figure 5.1: Our iterative design and development process.

5.1 Literature Study

As a first step towards our goal a literature study was held to find out what augmented reality applications exist today and what problems are common in these applications.

We looked for both articles regarding mobile augmented reality applications as a whole but also for very specific articles about the use of augmented reality for learning and instructions. We did this because we realized that instructional augmented reality applications often have problems in common with other types of augmented reality apps, even in some cases with geolocation augmented reality applications. User-evaluation articles were needed to know more about the users perspective of problems with using an augmented reality app. Articles about already developed augmented reality apps were used

to get an idea of problems other developers had noticed and solved before.

5.2 Brainstorming

After the literature study we had discussions to find out which problems were relevant for indoor and instructional AR applications.

During the rest of the project we had regular discussions and brainstorming sessions where we tried to come up with solutions to the challenges that we found in the initial literature study. These discussions were performed on the fly as we came up with ideas on how to solve the challenges. The ideas that we felt were good during these discussions would then be implemented and tested in our prototype application. If they did not fit they would be discussed and reworked or scrapped entirely.

We also had a few meetings with VSM Group AB. During these meetings we demonstrated the state of the application and also discussed how we were to proceed with the application and the thesis.

We chose this method as it was hard to know beforehand which solutions would work well in an augmented reality application and wanted to quickly be able to test new ideas. As well as being only two people in the project it was easy to start these discussions with each other.

5.3 Design and Development

For the development we used a method based on elements of Kanban. Kanban can be seen as a system which proposes different areas to consider when creating a development method. A key part of the system is visualization of workflow, which is done on a so called Kanban board. The Kanban board consists of rows where each row contains a user story, a priority and a status. A user story is a sentence of a requirement on the product from some user's perspective. The status can be either "Not Planned", "Planned", "In Progress" or "Done". We regularly went through the list and prioritized all the user stories that were not yet planned. This way new ideas were discussed and done in the prioritized order. [24]

Who	Goal	Reason	Priority	Status
Developer	To build foundations for a guide	Ease the construction of more guides later on	10	DONE
User	To have a guide for changing spools	Be able to get instructions on how to change the spool	25	DONE
User	To be able to freeze the camera	Have easier interaction when it is hard to hold the camera steady	20	DONE

Figure 5.2: Some examples of user stories that we had during the development.

The user stories can be read in the following way "As <Who> I want <Goal> to <Reason>". For example the second user story would read "As User I want To have a guide for changing spools to Be able to get instructions on how to change the spool". Formating the ideas into sentences like this makes it easier to understand the purpose of the idea.

The board makes it easy to see how far the development has come and also what the other developer is currently working with. It also makes it easy to suggest and discuss new ideas. New ideas were added to to our Kanban board with status "Not Planned" and when the number of planned tasks were starting to get low we discussed and prioritized the new ideas. Being only two developers and having full control over what to add to the application, we also allowed ourselves to sometimes re-prioritize tasks as we came up with new ideas so that good ideas did not have to wait a long time to be planned. [24]

5.4 Development Tools and APIs

5.4.1 Vuforia

Vuforia[1] is an augmented reality framework developed by Qualcomm. We chose it for several reasons. Vuforia seemed to be one of the best ones out there and on top of that it is free to use. Another reason was that Attentec had several people with prior knowledge of Vuforia which meant that we could get better supervising. It also has a Unity extension. We used Vuforia's Unity SDK version 2.8.7.

[1] https://www.vuforia.com/

5.4.2 Unity 4

Unity[2] is a 3D game engine in which you can build Android and iOS applications. Using such a software for AR development is good because it simplifies the graphical design when developing the 3D AR interface and let us focus more on the functionality of the interface. By using an editor like Unity it is easy to move around different AR objects and quickly see changes of how the interface looks. Version 4.3 was used.

5.4.3 Graphics

We used Blender[3] and Adobe Illustrator CC[4] to make graphics for the interface, and Paint.NET[5] to edit photos.

[2] https://unity3d.com/
[3] http://www.blender.org/
[4] https://creative.adobe.com/products/illustrator
[5] http://www.getpaint.net/

6 Result

In this chapter we present solutions to problems discussed in the Theory chapter and explain features implemented in the developed prototype application.

6.1 Detection and Tracking

The quality of the detection and tracking, determined by the factors described in sections 4.2.3 and 4.4.1, affects how frequently and quickly targets are detected and how well the interface is positioned. In some cases it is possible to detect targets almost instantly, giving a near seamless experience to the user. With other targets, or under worse conditions, the process may require much more patience.

6.1.1 Camera Management

Something we immediately noticed when testing was that camera focus had a significant impact on detection and tracking. Devices that do not have the ability to focus automatically – with continuous auto focus – have to trigger focus manually; for example through user input, or at a timed interval. The process of focusing is noticeable, the image becomes blurry for a short period before the camera is able to focus, and not desirable when the camera is already focused. Because of this, we decided to let the user trigger focus by clicking the screen, similar to how it works in the default camera app for Android [11]. Focus is only triggered when nothing else is clicked. If the user clicks an element of the interface, the focus should not trigger.

6.1.2 Target Selection

Below is an image of a machine, similar to the one we worked with, with potential targets highlighted:

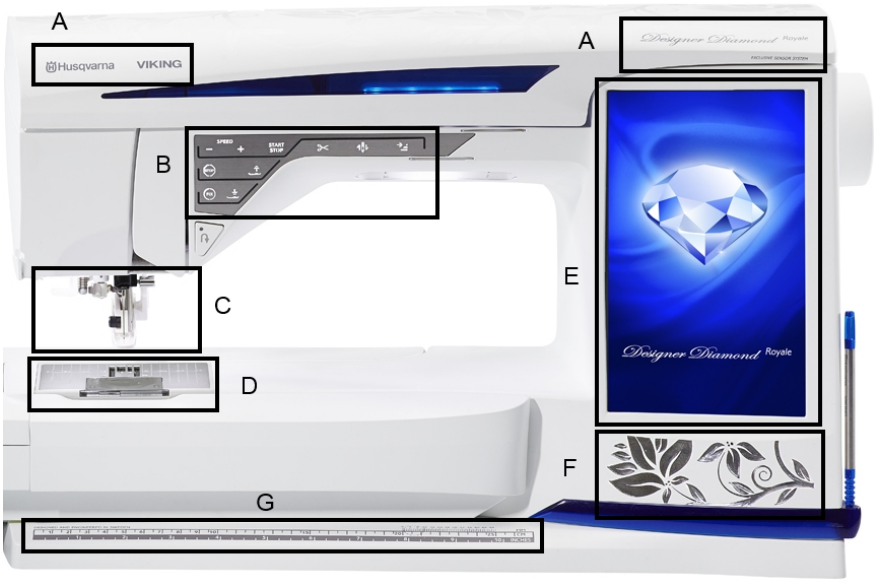

Figure 6.1: The Husqvarna Viking Designer Deluxe, with potential targets marked with letters A-G.

According to the designers of the machine the most interesting parts to annotate for the user are the needle area (C and D), the keyboard (B) and the screen (E).

Below is a list containing our evaluations of the potential targets, based on their suitability with regards to the factors described earlier.

A A logo is useful as a target because it looks almost the same on many different models. Unfortunately both logos are, in this case, placed quite a bit away from any of the points of interests. Even worse, they are placed on the lid, which can be opened. When the lid is opened, the logo will move and the relative position between it and the rest of the machine changes. Any augmentations on the rest of the machine that are placed with the logo as reference will either have to take that movement into account, or be placed wrongly when it is not in a specific position. Since there is nothing on the lid that is interesting to augment, we decided to not use the logos as targets.

B This keyboard is an excellent target because it contains a large amount of interesting details, both from the perspective of the

user and image recognition. It is also placed centrally on the machine. The text and graphics on the buttons are holes lit up by LEDs. When they are off it is almost impossible to trigger because the contrast to the background is greatly reduced. A benefit of the LEDs is that it should be possible to trigger even when the room is dark.

C This needle area is entirely three dimensional. The AR framework we chose does not support 3D tracking of arbitrary shapes. The angle from which it is viewed will therefore greatly affect how it looks, due to the depth, and thus whether it is recognizable or not. Because of this it is difficult to use it as a target.

D This metallic sewing plate has enough detail to be usable. The plate is engraved with numbers and lines that are detectable as features. The plate is lit up from above by a set of adjustable LED lights which when dimmed causes camera banding noise, likely having a negative effect on the tracking.

E Screens can be used to display an image marker target. In this case the screen is large and therefore very useful. Unfortunately, this would require the user to first put the machine in some sort of AR-mode that displays the target on the screen before using the app. You could also use images of the machine's own graphical user interface as targets, but in this case only a small portion of the interface is static, and you would need a very large amount of targets to cover all states of the interface.

F This flower painting does not exist on the model we used when testing. The painting appears to have a lot of details and potential features but unfortunately a reflective finish.

G The ruler might be a useful target, but it is quite far from anything interesting.

We primarily use **B** and **D** in our app. We have also had some success triggering on the screen and the logos, but chose to not use those targets due to their limitations described above.

We took several photos of the potential targets. These were cropped and edited to remove any parts of the image that could change, such as the background or any movable or changeable parts. Figures 6.2 and 6.3 shows two photos we used as targets for B and D.

Figure 6.2: The panel. The photo has been edited to follow the shape of the machine, cutting out parts of the photo that captured the room behind the machine.

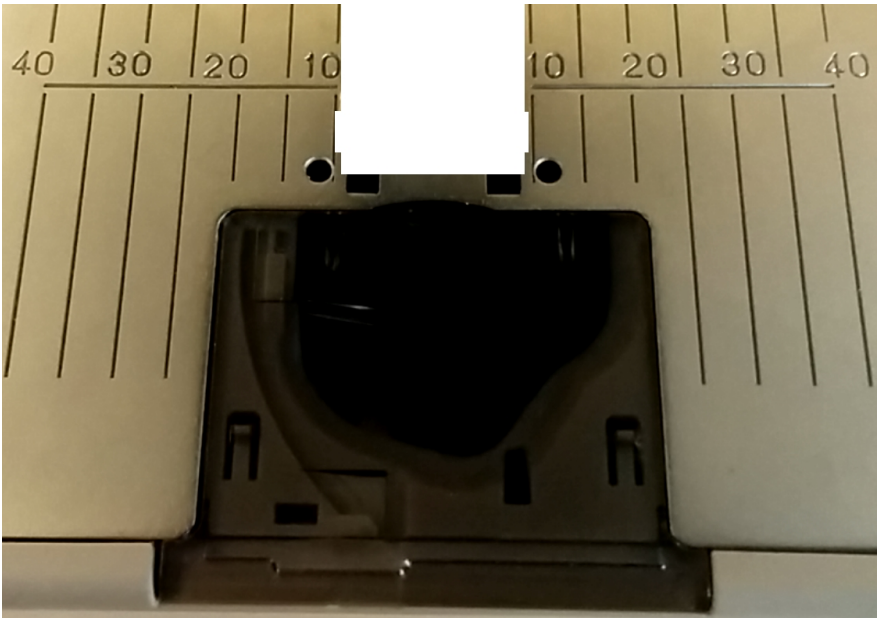

Figure 6.3: The plate. Note that the sewing foot, a part of the machine that can move or be replaced, has been edited out of the photo.

6.1.3 Target Management

We tested our app in different conditions and with different cameras. We discovered that triggering usually worked best when the condi-

tions were exactly the same as when the targets were created. It is possible to use several photos of the same natural feature, all taken with different cameras and in different lighting conditions. When this is done, the application can trigger on whichever is the best fit. This makes the detection less sensitive to lighting and camera changes.

To get the application to trigger on a target and show AR elements around it we used something called an *ImageTarget* which is a prefab, a kind of template, added to Unity by Vuforia. This prefab consists of a target image such as one of the target pictures above as well as some tracking settings. It also has references to child objects, which inherit the target's pose. These objects are shown when the application has triggered on the target, meaning that the children to the *ImageTarget* is our AR layer.

As we have multiple images for the same target as well as a couple of different targets we needed to use multiple *ImageTargets* and all of them should show the same AR layer. To not have multiple copies of the AR layer that needed to be in sync we developed a way that moves the AR layer so it is always the child to the currently tracked *ImageTarget*.

To implement this we created a class called *HandleARLayer* consisting of a reference to the AR layer and a *MoveARLayer* function that changes the parent of the layer. Then we made use of a callback *OnTrackingFound*, which is called each time an *ImageTarget* is tracked, to call the *MoveARLayer* function. This way we could make changes in one AR layer and it shows up the same regardless of which target is being tracked.

6.2 Development Using References

One of the most important aspects of augmented reality is the connection between the real and the virtual world. The layer of data that is displayed should be connected to what the user is seeing. When building a user manual, the information added with the app needs to be accurately placed, so that the user understands what part of the machine the information refers to and so that the information is accessible simply by looking at the right place.

By using a 3D model of the physical object, see figure 6.4 below, the designer can easily place elements of the AR layer at the right place by simply aligning it with the model.

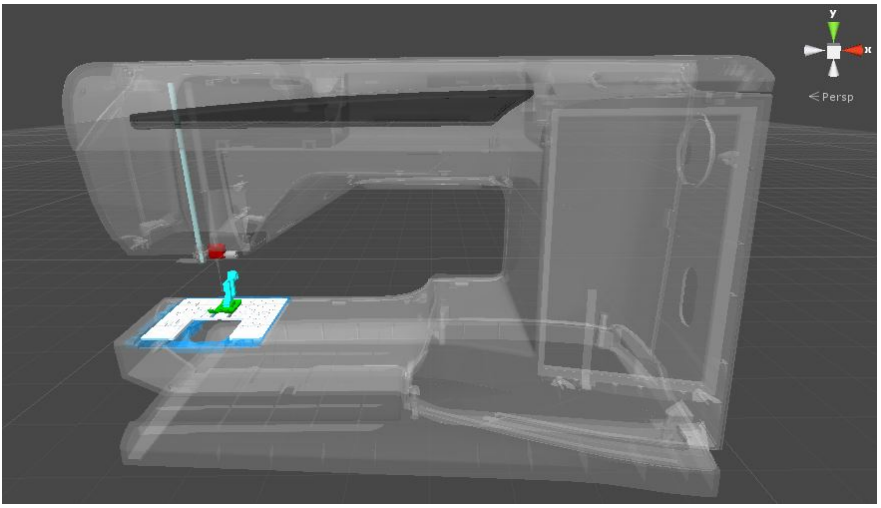

Figure 6.4: The 3D model we used while developing the app.

6.3 Generalization

Using references works great for finding where parts of the machine are located and to place elements correctly. However using a reference to place the elements in the AR interface means that it will only work for the specific machine that the 3D model is a reference of. Many products often have different models that still have the same type of functionality. This means that user manuals for multiple models often look the same and contain similar information. It is desirable when developing AR user manuals that the same information can be reused for multiple models so that the development time does not increase linearly with the number of models.

6.3.1 Coordinate and Information Databases

With multiple models in mind we implemented a couple of classes that handles information that should be able to change depending on the machine model. The first one is a coordinate database that maps names of machine parts to their position on the machine. The other database stores information and instruction strings for the machine parts. Other functionality in the app can ask for information or positions of a specific part via the database handlers instead of directly using information for one specific model.

As an example, practically all machines have a 'Start' button, but its placement and exact functionality may differ from model to model. When generating the AR interface from the databases, the coordinate database would be used to receive the button's coordinates on the model chosen by the user. If coordinates exists, we place an AR element accordingly and fill it with the information from the information database.

To initiate the databases and fill them with information of the machine parts there is still a lot of manual work to do. However, this has to be done only once for machine parts that are the same for multiple models. This is an improvement over having to manually add the information and place all the elements of the AR interface for each model.

6.3.2 Database Implementation

The implementation in our prototype uses JSON formatted files that store the coordinates and information and act as databases, see example in figure 6.5 below. At startup the database handlers parses the files and stores the information in dictionaries, mapping the name of the parts to their respective values. As we only had one machine to test on we did not add databases for other models. Only some of the features in the application uses the databases as the current information in them is limited. Future work on the application could be done to make it even more generalized.

```
[{
    "name" : NeedleHolder,
    "pos" : [-111, -9, 29]
},
{
    "name" : NeedleThreader,
    "pos" : [-130, -9, 29]
},
{
    "name" : Reverse,
    "pos" : [-100, 4, 10.5]
}]
```

Figure 6.5: Example of JSON elements in the coordinate database.

6.4 Interface

6.4.1 Robustness

To deal with the problem of interface stability, described in section 4.4.2, we tried to make sure that all clickable elements were large, far apart, and non-destructive. Larger buttons are easier to hit, and it is more difficult to click the wrong button if there is a distance between them. By making sure that no harm, such as deleting data, can be done by clicking a button in the AR interface, any misclicks are harmless.

The error of the placement is most noticeable far from the active target. This means that relying on accurate placement, for instance when annotating, can be problematic if the distance to the nearest target is too large. The interface can easily be erroneously placed or rotated so that an element does not appear where the designer expects it to be. Exact placement should therefore probably not be crucial to the use of the interface, at least not when the active target is far away from the placed item.

6.4.2 Scaling and Interface Legibility

With the movement of the camera in mind and frequent testing from various angles and distances we arrived at the following solution to the problem of interface scaling:

Less detailed but larger information is shown from afar or from large angles, while smaller elements with more detail is shown up close. The lower detailed elements fade out as the camera approaches to avoid cluttering, and reappear when the camera moves or looks away. See figure 6.6 below.

Many elements of the interface, such as text and images, are inherently two dimensional. Such elements can be rotated to always face the camera, but this reduces the sense that they exist in the real world. The placement and scale of the interface can instead be used to encourage the user to move the camera. By using the technique described above, objects that require precise placement can be shown only when we know that the user is pointing the camera directly at the target. This effectively mitigates the problem of uncertain placement far from targets, discussed in section 4.4.2.

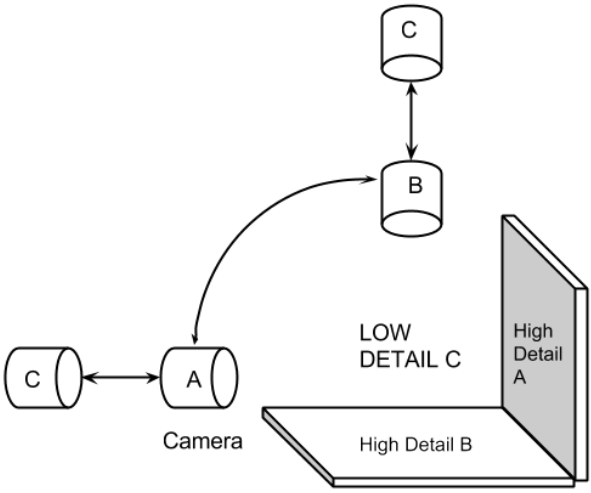

Figure 6.6: Different camera positions and detail layers.

As seen in figure 6.6 above, different elements can be shown depending on the camera's position. Low detail is shown from afar, in position C, and higher detail from the two closer positions A and B. Depending on the angle, the two higher detail layers can fade in and out to ensure that they are not in the way of the line of sight. When the camera is looking at "High Detail A", the info in "High Detail B" is hidden so that it does not get in the way. The orientation of the camera and the object (their forward vectors) are used to calculate that angle, see figure 6.7 below.

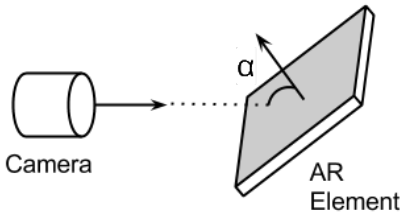

Figure 6.7: How the angle is calculated.

The switch between low and high detail can be done gradually or all at once at a threshold point. When done gradually, we think the user gets a greater sense of control. The instant response to the camera movement makes it easier for the user to learn that the camera's movement is connected to the fading of the layers. If the fade triggers at an arbitrary threshold point, it might be more difficult for the user to understand why it happened.

Using Detail Layers

In our app, an implementation of this is used in several places. A template was created which lets the developer simply drag elements into high or low detail groups, then it is possible to adjust parameters for the distances and angles that control the fade as well as the animations themselves. The figures 6.8 to 6.10 below show one example of how it is used.

Figure 6.8: This shows the lowest detail level.

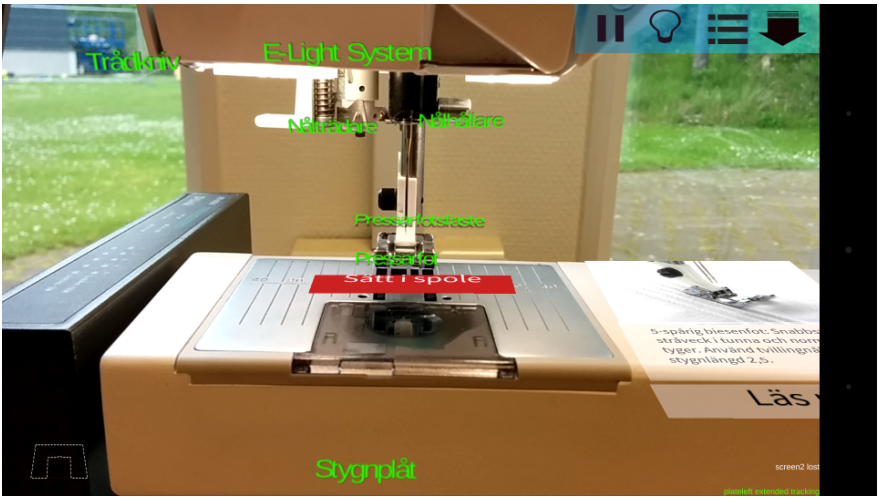

Figure 6.9: When the camera moves closer, the large text in the previous image (figure 6.8) fades away and smaller texts are shown instead.

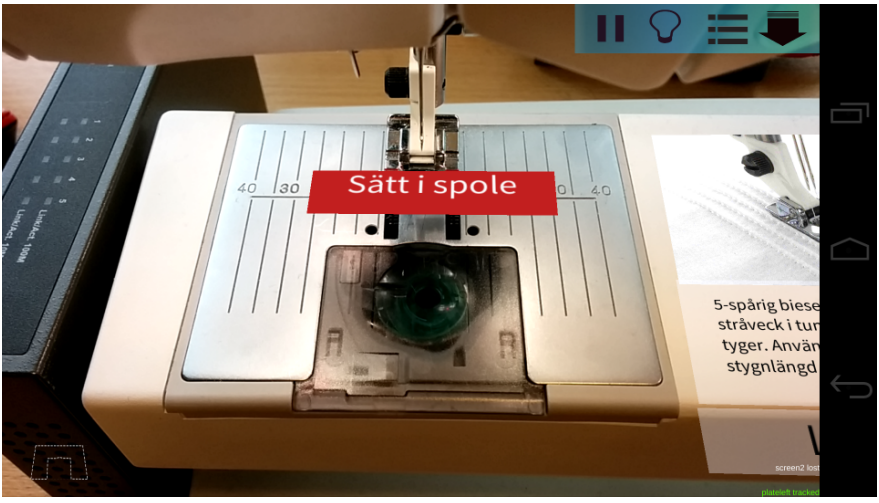

Figure 6.10: This shows a level where the text in the previous figure (6.9) has been hidden due to the angle from which it is viewed.

In this case we are using two nested detail layers. In the second image, figure 6.9, when the camera is moved backwards the outer detail layer

will fade to a lower detail level, showing only the large green text in the first image, figure 6.8. The small green texts facing the camera will fade away when the camera is moved to look at the machine from above, as is seen in the third image, figure 6.10. These layers are there for two reasons. Firstly, we do not want to show the text seen in figure 6.9 when the user is looking at the guide in figure 6.10. It is not useful from that angle, and may be in the way of the guide's contents. Secondly, we want to be able to control where the user views the machine from. The detail layers prevent the user from moving too far away from a target and from viewing it from a bad angle.

6.4.3 Text Legibility

In AR applications like ours, the camera feed usually acts as the app's background. When displaying text in the interface over the camera image, it is sometimes difficult to read the text if its color is similar to the background. To fix this, we have added black shadows to all texts, as this was shown to have a low response time in a study on text legibility by Gabbard and Swan [8]. The shadows make the text stand out more, diminishing the effect of the background. We also often add a transparent background behind the text to increase the contrast.

We also attempted another solution to the problem. We tested using a custom shader, a small program that describes how to render the text on the screen, that would draw the text in the "opposite" color of the background. If the background is black, the text should be drawn white. Unfortunately we were unable to write a shader that handled gray backgrounds well. Our shader simply subtracted the background color from white. It can be thought of mathematically as the function y = 1 - x, where x, the background color, is a value between 0 (black) and 1 (white). Subtracting white from white yields black (1 - 1 = 0), but subtracting gray from white results in putting a gray colored text over a gray background (1 - 0.5 = 0.5), which is the opposite of what we wanted, see the figure 6.11 below.

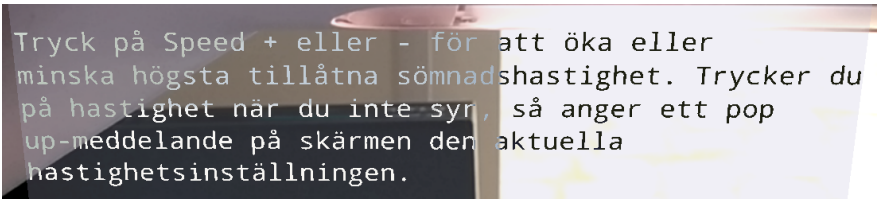
Tryck på Speed + eller - för att öka eller
minska högsta tillåtna sömnadshastighet. Trycker du
på hastighet när du inte syr, så anger ett pop
up-meddelande på skärmen den aktuella
hastighetsinställningen.

Figure 6.11: Augmented Reality 3D text with dynamic color based on the background. Note the black text on white to the right, the white text on black in the lower left corner, and the gray text on gray in the upper left corner.

As our test machine itself is gray we decided not to use our shader solution and instead settled on having just solid color text with a subtle shadow, see figure 6.12 below.

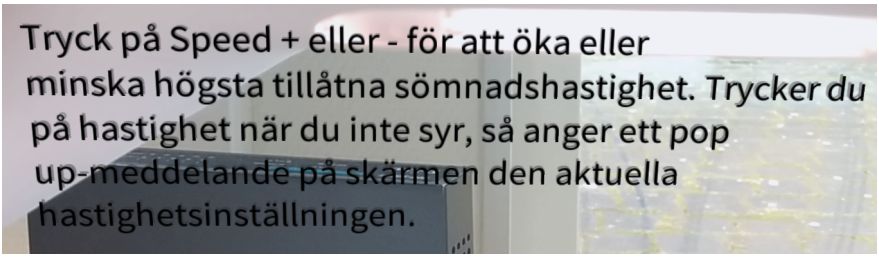
Tryck på Speed + eller - för att öka eller
minska högsta tillåtna sömnadshastighet. Trycker du
på hastighet när du inte syr, så anger ett pop
up-meddelande på skärmen den aktuella
hastighetsinställningen.

Figure 6.12: Augmented Reality text, colored black with a subtle shadow and white transparent background.

6.4.4 User Tutorial

Mobile AR being a new technology means that AR applications will often be the very first AR experience for many users and, as discussed in section 4.3, it is not easy for the user to know how to use such an application for the first time. Therefore there is a need to explain to the user how to use the application. Perhaps even more so in vision-based applications like ours, as they are generally more complex to use than sensor-based applications. For example, a user manual app requires the app to trigger on a specific target, which requires good lighting and that the user directs the camera at the right places. Whereas many sensor based applications only need the user to hold the camera in any direction.

Some of the things our application needs to teach the user are where

on the machine the user can trigger, how the camera can be moved when the application has triggered and that the lighting in the room can affect the performance.

Implementation

To teach the user how to properly use the app, the app first displays a screen with the most basic instructions seen in figure 6.13. This screen tells the user to make sure the room is well lit and to point the camera in the direction of the available targets.

Så här använder du applikationen:
Rikta kameran mot en av följande delar tills du ser att 3D gränssnittet dyker upp.
Se till att rummet du befinner dig i är väl upplyst
Tryck på skärmen för att stänga denna instruktion

Figure 6.13: Introduction screen with explanations of how to use the applications.

When the user closes the intro screen, an outline of one of the targets appears, see figure 6.14. The outline helps the user to align the camera correctly. When the camera is held at the right distance and angle, the outline should match the target perfectly. The outlines are constructed manually from the images used to create targets, so that the camera will see the target as it looks in the photo when the outline is correctly aligned.

When the outline is aligned correctly, the user is holding the camera in an optimal position for triggering. If applicable, we also display the text "tap screen to focus" in this mode to teach the user about the camera focus method (see section 6.1.1). When the app triggers on

the target the outline disappears and the actual AR layer appears instead.

Figure 6.14: Outline meant to help the user to find a good position for triggering.

The detail layers described in section 6.4.2 are used to pull the user toward the targets, helping them learn where and how to hold the camera. However, this will only work once the user has managed to trigger and when tracking is working.

We wanted the app to be at least somewhat usable even without triggering, mainly because triggering can be difficult and frustrating. To achieve this we designed the GUI so that it provides help for the user and includes features that may be useful even without AR.

6.4.5 GUI

We designed and implemented a non-AR GUI. We decided that some elements of the app must be available to the user before triggering, such as the ability to toggle the flash on or off. We also decided that elements that do not have any direct connection to a place in the real world – no obvious place to be in the AR layer – should be in the GUI instead.

The GUI has a special role in AR applications in that it is the first and only part of the app the user sees before triggering. As we have mentioned before, vision-based AR is mostly useless if the user

is unable to trigger. By moving features that do not really need to be shown in AR to the GUI, more functionality is immediately available to the user when starting the app.

The main menu is kept as small as possible, partially transparent and out of the way from what is shown on the camera. During development we iteratively made the menu smaller to give as much space as possible to the camera view. Through the main menu it is possible to open other menus, the guide view, the index and the guide list, which slide in from the edge of the screen. These were also designed to be as small as possible while still remaining functional. See figure 6.15 below.

Figure 6.15: The GUI (left, within the gray frame) in a 16:9 aspect ratio with all menus open. The Guide List (right) can replace the Index in the position below the main menu.

Only the main menu is always shown. The Guide View is enabled when a guide is started from the Guide List. The Index or the Guide List are opened or closed by clicking the index button (second from right) or guide button (first from right) in the main menu. The texts in the GUI Guide View are placeholders that are replaced when a guide is started. The buttons in the guide list above are mostly placeholders to show how the menu looks when it is filled out. How the guides work is shown later in section 6.8.

The Outline Menu manually toggles a full size version of the outline when pressed.

6.5 Interaction

6.5.1 Embodied Touch Interaction

To make it possible to interact with elements of our AR interface, we wanted to utilize the mobile device's touchscreens to make them clickable.

To do this, we implemented a solution using raycasting. When the user presses the screen the application checks if the coordinate from the touch event hits any element of the GUI. If that is not the case, a ray is sent from the pressed coordinate in the direction the camera is looking. The first element it collides with, if any, receives a click event. The ray stops when it collides with a clickable element or after it has traveled a certain maximum distance, which is set so that any elements that look clickable are within reach of the ray.

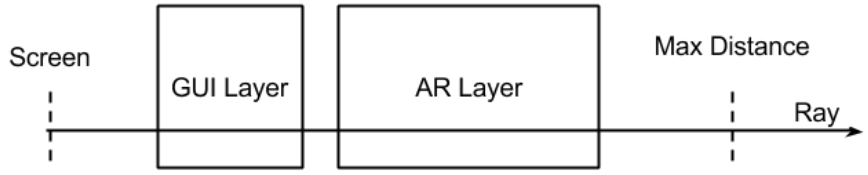

Figure 6.16: This illustration shows how clicks are handled.

For the implementation of the touch handling we created an interface, *ITouchable*, which all touchable elements must implement. The interface contains the methods *OnTouchDown* and *OnTouchUp* that are called when the element is touched. If the ray collides with an element that does not implement the *ITouchable* interface, it passes through and can collide with other elements behind it. If no element is pressed, manual camera focus is triggered.

6.5.2 Tangible Interaction with Virtual Buttons

Vuforia supports a method called Virtual Buttons, described in detail in the theory section 4.3.2. Virtual buttons allows the user's physical interactions with targets to be detected and used as button presses.

While using virtual buttons is cool in concept it relies a lot on that the tracking conditions are very good and is therefore a questionable solution in an instructional AR application meant for use at a users own home where conditions may vary a lot.

When we tested virtual buttons they worked very poorly. Even at the lowest sensitivity settings, they would trigger seemingly without reason. You must also take great care in placing and sizing these buttons, as you easily 'press' the wrong button simply by moving your finger over it, for instance when trying to press another button.

Lastly, we felt that since we would not be able to use virtual buttons consistently throughout the entire interface, it was best to not use them at all as doing so could be confusing for the user.

6.5.3 Ability to Freeze the Display

In section 4.4.4 we explained that the ability to freeze the display can be a good solution for problems with interaction while holding the device. This functionality was implemented in our prototype, see figure 6.17 below.

Figure 6.17: The application in use in a freezed state.

The application can at any time be put into freeze mode by pressing the pause button in the upper right hand corner.

In this frozen state the application can still be interacted with, as

seen in figure 6.17 above. The step by step instructions on the screen can be browsed and the steps will be animated just like it would be if the application were in its normal state when it is showing the live camera as the background.

The actual implementation of the freeze mode is very simple when using Vuforia. Most of the implementation is just a call to a pause method in the Vuforia class that handles rendering of the background, *QCARRenderer*. To get the application to maintain the AR interface in the frozen state even if the application was minimized, for example by using the home button on Android, and then started again we had to override Vuforia's behavior and let the AR interface be in an active state until the user have focus on the application again.

6.6 Machine Overview

A machine overview is something that exists in most user manuals. Its purpose is both to tell the user what parts exists on the machine but also to teach the user the name of the part so that it can be looked up in the index for more information. A machine overview is something that in theory works very well with AR since we can show what exists on the user's machine. In practice there are, as with many things in AR, some complications that limits its use, but with good tracking conditions it can work really well. Below in figure 6.18 are an image of the machine overview in the original paper manual and in the figures 6.19 and 6.20 are two images from the version in our AR manual application.

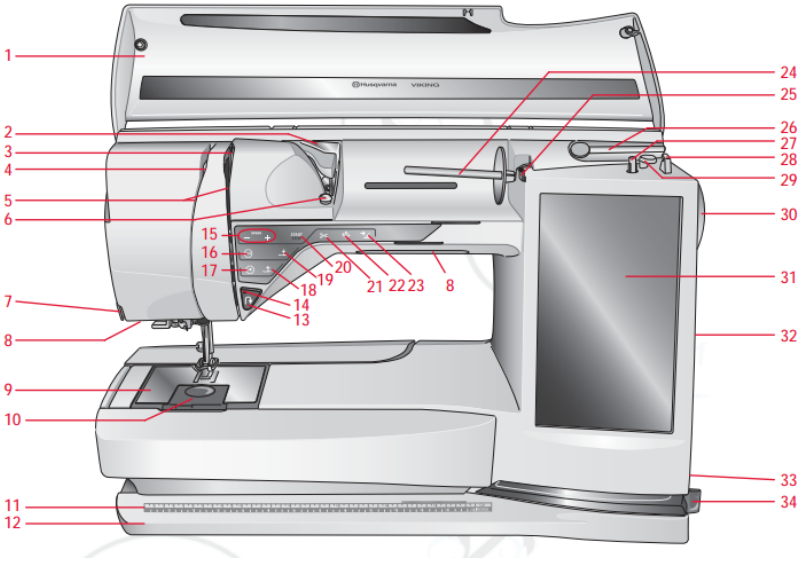

Figure 6.18: Machine overview from the user manual of Husqvarna Viking Designer Diamond. On the same page is a list explaining what each number refers to.

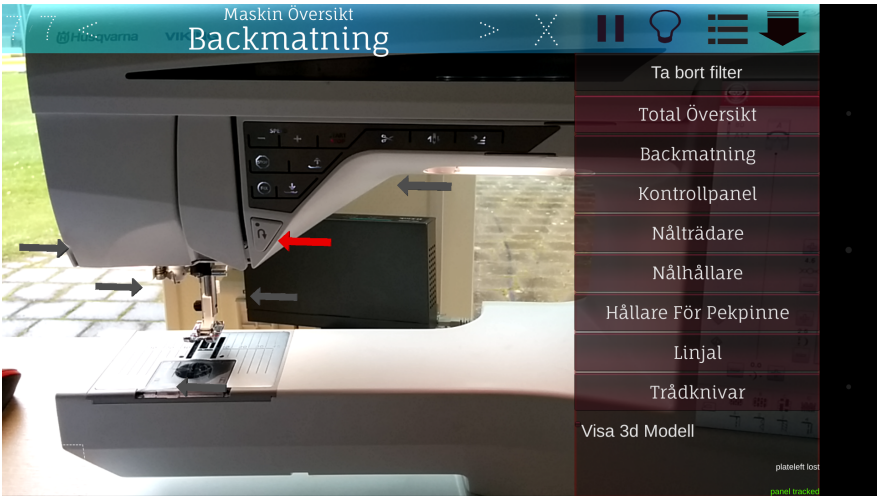

Figure 6.19: Machine overview from our app, with the 'Backmatning' arrow selected and colored red. The arrows are clickable.

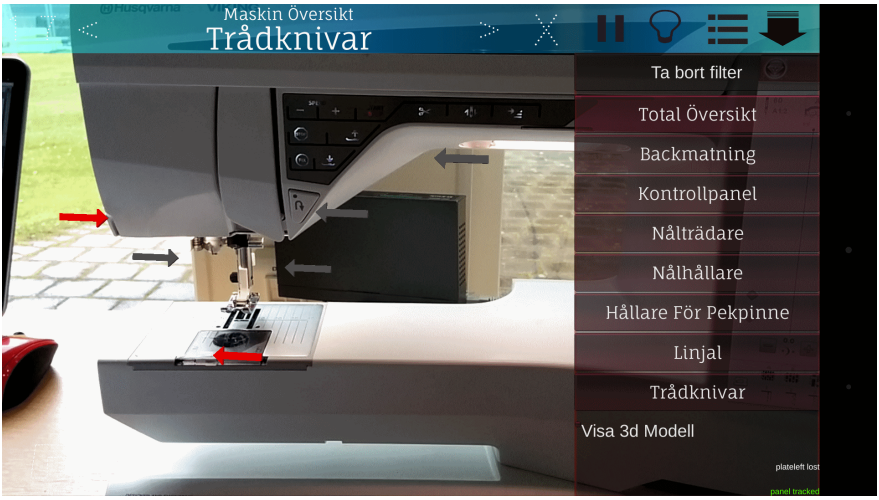

Figure 6.20: Machine overview from our app, showing the locations where thread knives can be found on the machine.

6.6.1 Use of the Machine Overview in AR

The machine overview is reached via the menu icon in the upper right corner. By clicking on the button "Total Översikt", total overview, the arrows in figures 6.19 and 6.20 above are displayed on the screen and a guide with steps naming all of the parts the arrows are pointing at is shown. The user can go through the steps either by using the guide in the GUI or by directly clicking the arrows on the machine. When an arrow is clicked or the corresponding step in the guide is shown the arrow is displayed in red and the name of the part is shown in the guide.

As can be seen in the pictures above there are also buttons in the index menu for different parts of the machine. This allows for more uses of the application as the user may know the name of a part of the machine but not where it is located. Clicking on the button gives this information directly by just showing the arrows that point to that part of the machine. If the user knows the location of the part but not what its called the user can use the total machine overview and click the arrow pointing to the specific part of the machine to get the name of this part.

6.6.2 Implementation of Machine Overview in AR

The implementation of the machine overview makes use of the coordinate database that was discussed in section 6.3. The class that is responsible for creating the overview stores names of all the parts that should be visible in the overview and then hands these names over to the coordinate database and gets the positions in return. The overview class then creates arrows at all these positions via another helper class. The overview is therefore entirely model independent and by switching out the coordinate database to another model's database the overview would automatically work as intended.

6.7 Information Viewer

The information viewer, shown in figure 6.21 below, shows information about parts of the machine.

Trådklipp
Tryck på det automatiska trådklippet så skär maskinen av över- och undertråden, höjer pressarfoten och nålen samt aktiverar FIX-funktionen för nästa söm. Tryck på det automatiska trådklippet medan du syr för att skära av trådarna i slutet av en söm eller ett sömprogram.

Figure 6.21: The information view, showing information about the machine's keyboard.

Transparent white rectangular buttons were placed so that they are displayed over the physical buttons to indicate that the area is clickable in the app. When the user presses a button, it is selected. The selected button fades to a green tint and information about the physical button behind it is shown in the viewer. The stationary viewer is placed within view so that the text is readable without moving the camera away from the buttons.

We experimented with various solutions before this one. First, text was shown for all elements at once. With that system, it was impossible to show much text due to the limited space available. We

then introduced the buttons. First, the buttons were colored areas with text on them. However, we felt that the text was unnecessary as there is already text and images on the physical machine. By using transparent buttons, the machine is visible behind them.

6.7.1 Implementation

Two classes are used to implement this system: the *InformationViewer* and the *InformationButton*. The *InformationViewer* stores the selected button and makes sure to display the text that the button represents. The *InformationButton* class, which implements the *ITouchable* interface mentioned in section 6.5.1, contains references to the texts that should be displayed when the button is selected and a reference to an associated *InformationViewer* instance. The texts stored for each button are fetched at startup from the information database discussed in section 6.3, which means that the texts can easily be changed depending on which database file is in use. When the *InformationButton* receives a click event, through the *ITouchable* interface, it calls a method in the *InformationViewer* instance telling it to select that button. The viewer also deselects the previously selected button and updates the texts.

6.8 Guide Systems

Guides are longer sets of instructions, often divided into multiple steps, designed to teach the user how to accomplish a task.

We have implemented two kinds of guides: detailed, animated guides in the AR layer called *AR Guides*, and text-based guides in the GUI called *GUI Guides*.

6.8.1 AR Guides

The AR guides are used for specific instructions that are displayed directly on top of the machine. The instructions can be anything from simple arrows or images to animations showing more advanced operations.

A guide is started by first finding it in the AR interface and then clicking on the title of the guide. 'Next' and 'Previous' buttons are shown so the user can go through all the steps in the guide.

We designed a couple of guides, both based on guides found in the printed user manual for the product. One of the guides covers installation of the bobbin, it has the following four steps:

1. Remove the lid that covers the bobbin container.
2. Place the new bobbin into the container.
3. Wind the bobbin thread.
4. Put the lid back on.

All of the steps are animated to show exactly how they are performed on the machine. See screenshots from our implementation in figures 6.22, 6.23 and 6.24 below.

Figure 6.22: The button to start the guide.

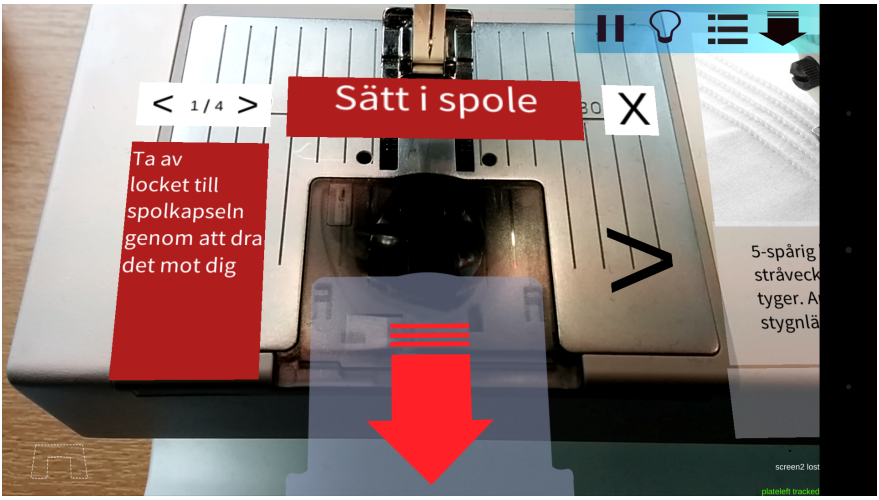

Figure 6.23: The first step of the guide, showing how to remove the lid with a repeating animation.

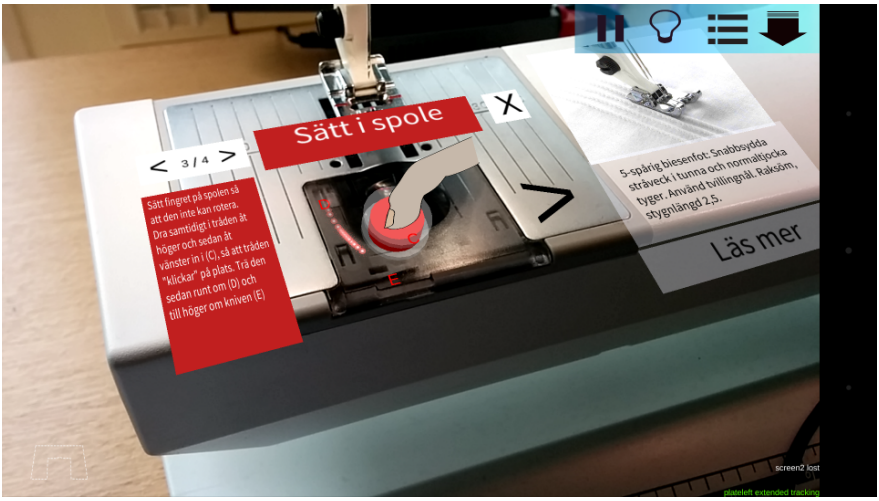

Figure 6.24: The third step, using animated dots to show how to wind the thread and an image showing that the bobbin should be held steady.

6.8.2 GUI Guides

The GUI guides are larger in scope, often covering multiple areas of the machine, teaching higher level tasks. Each step of these guides contains a brief description of what to do and, if appropriate, an arrow in the AR layer showing where it should be done. For instance, the guide might instruct the user to press a button, or tell them to switch to a different type of sewing foot. In the latter case, the GUI guide would point to the AR guide which tells the user how to change sewing feet.

The GUI guides also make use of the outline help system. Since GUI guides can be used even when the app is not tracking a target, we can display the outline to help the user trigger on whichever target happens to be the best for the active guide step.

The rationale behind displaying these guides in the GUI rather than in the AR is that when implementing an AR guide that moves across the machine, its interface would either have to move along with the guide or stay in one particular place. A moving interface may confuse the user, especially if it is difficult to find when it moves out of view. With a static interface, the user might have to move the camera unnecessarily from point to point when following the guide. The movement also means that they seldom have an obvious starting point. We believe it is easier for the user to find these guides in an alphabetical list in the GUI, than it would be if they were placed on the machine.

We implemented an example guide which tells the user how to sew buttons. The guide shows which buttons to press, and tells the user to install the special sewing foot, pointing at the foot installation AR guide. See figure 6.25 below.

1 / 5 <
Sy knapp
Höj pressarfoten
>
X
Pressarfotslyft
Tryck på det automatiska pressarfotslyftet för att höja pressarfoten. Tryck på knappen igen så höjs pressarfoten till det extra lyft-läget och matartänderna sänks automatiskt så att det går lättare att föra in tjockt tyg under pressarfoten

Figure 6.25: A step from a GUI guide for sewing buttons, telling the user to press a button (to the right of the big arrow).

6.8.3 Implementation of the Guides

Both guides have very similar implementations. They consist of a guide handler class that controls the flow of the guide and contains functions to start or end the guide and switching between steps. The difference between the guide handlers is that the *ARGuideHandler* class controls and stores a list of *ARGuideStep* classes that stores a 3D model or text that exist in AR while the *GUIGuideHandler* has a list of *GuideStep* classes that represents texts that should be displayed in the GUI. While both the *ARGuideStep*'s and *GuideStep*'s main purpose is to store the information that should be displayed in that step the *ARGuideStep* class also has logic for how to hide and show the step.

In order to make guides work with multiple different models, using the databases described in 6.3, we wanted to separate the actual content of the guides from the visualization. This was easy to do for the GUI Guides, where the instructions and locations could easily be fetched from the databases. The AR guides, however, have a large amount of elements used to animate the steps that have to be placed with accuracy, making the use of databases more difficult.

7 Discussion

7.1 Method

7.1.1 Iterative approach

Early on we had a far less iterative approach in mind. The plan was to first design the app and then implement it. When we started to think about and design the app on paper and with wireframes, we detected that this process did not suit the project. When designing on paper it was incredibly difficult to guess what would work well, especially with AR. We had no prior experience working with Vuforia, or indeed AR at all, so for us it was important to test our ideas early by developing small prototypes. During and after development of such ideas, we would think about their benefits and potential drawbacks from different perspectives. We tried to use the new information to improve the entire app. Very rarely did our original idea remain unchanged.

As an example, our earliest design of the detail layers was very different from the final version described in 6.4.2. Instead of zooming between layers by moving the camera, the user would click a button in the center of the area to toggle between the lower and higher detail levels. This seemed like a good idea when drawn on paper. It was not until we tested a rough implementation of the idea that we realized that camera movement would be a much more natural input for the problem than clicking the screen. Discovering that solution without testing was difficult for us, primarily because almost all software we use is two dimensional. We are not used to the addition of the third dimension to the interface. It was also because, without experience with the platform, we did not know what was easy, difficult or impossible to do from a technical point of view. In this case, we are using the distance and angle between the virtual camera and the center of a group of objects as input, and the opacity of the elements of that group as output. Before becoming familiar with the framework, it is not obvious that such values are easily obtainable.

A drawback with this approach is perhaps that we never really spend enough time planning ahead, which may result in an end prod-

uct that lacks cohesion. However, it was never intended that the product of this thesis should be a finished, commercial product. The app is only a prototype showing several examples of what you could and should do when developing similar apps. Thus any lack of cohesion and polish is mostly irrelevant.

7.1.2 Brainstorming

Our informal brainstorming mainly included thinking out loud, with occasional note-taking. We looked at articles, apps and our own ideas and implementations, trying to generate as many ideas as possible by discussing them from different points of view.

More discussions and interviews with other people, like potential users of the app and experts in AR, would perhaps have generated ideas that we missed. We did discuss the app with the developers of the machine that the app targeted. These discussions did lead to a few good ideas as well as information about which aspects of the app they were most interested in.

7.1.3 Kanban

The decision to use Kanban was made a few days after development started. We wanted an incredibly lightweight methodology with low overhead, and chose to utilize elements from Kanban based on our earlier experience with it.

We felt that Kanban suited our project very well. The Kanban board served as an extended to-do-list, with more information to make it easier for us to decide what to work on next. Because of this, it was easy to add ideas from our discussions to the Kanban board. The restrictions Kanban puts on the amount of simultaneously active tasks made sure that all tasks on the board had to be small, independent pieces that one developer could finish quickly, which perfectly suited our iterative approach.

We felt more effective while using Kanban, possibly because of the visual aspects of it, seeing tasks move from 'Planned' to 'In Progress' and finally reach 'Done'.

7.1.4 Use of Vuforia and Unity

Our main priorities when choosing the frameworks were that it should be easy for us to get started and make progress quickly. We were also

concerned about possible limitations that would force us to either abandon various ideas or change framework. In the end we chose Vuforia based on what we have heard from our supervisors at Attentec and what we have read online on websites for developers.

We have had almost no negative experiences with Vuforia during our development, although the tracking turned out to be a bottleneck for usability in many cases. Although it is possible that our prototype might have worked better if we used another framework, we were still able to achieve what we wanted. We do wish that it was open-source, however, as that would have allowed us to modify and get a greater understanding of the framework.

Working with Unity has several notable benefits over development with just the Android SDK. Of significant note is the visual, what-you-see-is-what-you-get editor of the app's 3D environment which enables quick design changes to be made without repeatedly compiling and running the app. This made our work a lot quicker and more precise. The ability to run the app on the computer, rather than having to build to a mobile device also saved time. The benefits of this were most notable in the beginning of the project. Unity also allows edits to be made while running, which has also been useful. The shift from code to Unity-format data files has the unfortunate side effect of making version control merging, with Git[1] in our case, tedious as those files could not easily be merged automatically. This could possibly have been circumvented by purchasing the Pro edition of Unity which promises better support for version control.

7.2 Result

7.2.1 Quality of Tracking

What affects the application's quality and overall usability the most is probably the quality of the natural feature tracking. With bad tracking the application will not be usable at all as the AR interface will not be correctly displayed.

With Vuforia we have been able to at least get the application to reach a middle ground. The application works and it is possible to demonstrate its features in a good way. However, it is hard to see the application being used easily by a normal user in its current state. For example, the tracking is good enough for the bobbin guide and the

[1] http://git-scm.com/

information view, but in both of these features the information are shown right on top of the actual targets. For the machine overview the information is spread out all over the machine. We cannot see the whole machine when triggering so the user first has to get close to the machine to trigger and then move the camera further back to get an overview. As the user gets further out however the tracking gets less precise and the overview may be displaced. The tracking is thus not good enough for the overview to be a reliable feature. What made it at least possible to an extent was Vuforia's extended tracking feature.

Extended tracking adds much to an AR application such as the one described in this thesis. Without it the tracking has to be much better or more time has to be spent on finding good image targets giving less time to spend on actual development. Features that require broad views, such as the machine overview, benefits a lot from extended tracking and would be near impossible to have if extended tracking did not exist.

7.2.2 Measuring the Quality of Targets

There are many factors that affect how well detection and tracking work. Because of the nature of these factors, as well as the sheer amount of them, it is incredibly difficult to see what effect different changes has had to one of those factors. When testing two different targets, the fact that one seems to work better than the other may not have much to do with the quality of the targets. Other factors such as the lighting in the room, the camera quality and the angle at which the camera is held when testing, are hard to replicate between tests and they can not be ignored. In the end, instead of trying to find a perfect target that could work in many different conditions we found that having multiple targets with different conditions worked well in our tests. However, we do not know if the application would have been more efficient if we were to find and use just one good universal target.

7.2.3 Use of Augmented Reality

Using augmented reality in an application can serve a couple of different purposes. AR can be a help for the user to easier understand information and in what context the information is relevant. But in many cases it also serves as making the application more fun and immersive. It is very easy to overuse AR and use it in places where it

does not serve any purpose and just gets in the way. For every idea we had during development we tried to think about which purpose AR served for that idea.

The freeze functionality is one example where we considered if it would be good or bad for the application. It is actually mentioned by Bai et al. that freezing the view does remove some of the fun and engaging feeling from the application as it is not real-time AR anymore and instead just a static image background. However, for an instructional AR application fun should not be the deciding factor. The freeze functionality serves a purpose as it lets the user see the instructions without holding the device which we felt was much more important. We also noticed during development of the application that this feature was good. For example, during demonstrations we were able to freeze the display in different areas of the application and hand the mobile device to the audience so they could interact with the application from their seat. Similar situations can be imagined during actual use of the application where one person triggers the interface and then wants to show the instructions to a different user. With freeze this can easily be accomplished.

7.2.4 Detail Layers

The poor quality of the tracking can be dealt with, at least to some degree, by using detail layers to force the user to stay within predefined zones where we know tracking works well. The user may not be aware of which parts of the machine the camera needs to face in order for the tracking to work. By telling the user explicitly to point the camera at the targets, through the outlines and the introduction screen, we get the user to the right positions. Since the high detail information fades away when the user moves away, we encourage the user to stay in those areas, while also hiding most of the ugly errors that could appear when the user is too far away.

The detail layers may have negative effects as well. The parameters that affect the allowed ranges and angles for the high detail level have a big impact as they effectively control where the user can view the interface from. If the allowed space is too small, the user may be forced to hold the camera in an uncomfortable position in order to see the high detail layer. The detail layers take some control away from the user, which may be frustrating. However, although we never experienced these drawbacks when testing, we have considered

and decided against alternatives like controlling which layer is visible through other input methods, like buttons, or abandoning the use of layers entirely.

One of the main benefits of detail layers is that the increased control for the designer makes it possible to decrease the risk of cluttering the interface. However, it is possible that it is insufficient in doing so, with a fully generalized approach to generating the interface and a more detailed interface. Then, a solution with dynamic placement of the AR elements, such as the ones discussed by Azuma and Furmanski (see section 4.4.3), may be necessary to reduce the cluttering. If a solution where occlusions could be detected in real-time was implemented, one could dynamically hide and show elements of the interface in a more intelligent manner, perhaps reducing the importance of fine adjustments by the designers.

7.2.5 AR Apps as User Manuals

User manual applications enhanced with AR definitely have some advantages over ordinary digital and physical user manuals. Exactly how big these advantages are and if they outweigh the drawbacks is hard to say.

One undeniable gain of augmented reality is the ability to place information so that the position of it is meaningful. If you were to open a physical user manual, you would have to look at the index to find out where information about something is placed because the placement is arbitrary. You cannot guess what page the information about a button is on. With augmented reality the placement of information makes sense and the user should be able to guess where to look.

Due to the limitations of tracking at larger ranges, it is difficult to achieve the same effect that is found in the overview in physical manuals in an AR app. Generally, one of the biggest issues with augmented reality is the imperfect tracking. Any time that the camera or the augmentation is too far away from the nearest good target, the precision of placements will suffer and the user experience with it. It is questionable whether users will have the patience to deal with those problems.

Visualizing instructions helps the user understand what to do. Using AR to show the instruction in 3D on the user's own machine seems more effective than a two dimensional animation, and certainly more effective than a set of static images. However, if the tasks taught by

the guides are too simple, text or images are enough. In that case, the use of AR is pointless since it takes more time to start the app and trigger on the machine than it takes to look up the information elsewhere.

At some point, the gain of AR is exceeded by the costs of its flaws. In the worst case scenario, augmented reality is unusable because it is impossible to guarantee that it even works. Developers should take into consideration that the process of starting the AR app can take anywhere from a couple of seconds to half a minute or more, and that even when it does work it can be unreliable and frustrating to use. All of this is mainly due to the poor quality of tracking. With even more powerful phones and better tracking methods many issues can be solved. The good news is that key tracking features have been added to leading AR frameworks, like Vuforia, recently, and one would expect more is to come in the future.

8 Conclusions

8.1 How can a mobile application use augmented reality to show how a certain tool is used?

Mobile applications can advantageously use AR for instructional applications, such as user manuals, by using 2D natural feature targets to recognize and track a machine in front of the user.

In this thesis we have shown several ways that AR can be used for instructional features. Animated guides placed directly on top of the machine show instructions in a very intuitive way, while more traditional GUI guides can teach higher level tasks consisting of AR guides and other pointers. For more general help the application can show information accessed via augmented reality buttons placed on top of actual buttons on the machine. It can also use a machine overview that shows the locations of the machine's parts. These features enables easy access to information about the machine even if the user does not know the names or positions of the parts.

By building the application in a way that enables coordinates and information about different parts to be easily switched out, the application can more easily be made to work with many models of a specific machine.

Developers should however take great care of what features are developed using AR. If AR is used for features that do not gain anything from it, the application will just get more complex and harder to use.

8.2 What can be done to improve the usability of instructional augmented reality applications?

We have identified and solved several issues affecting the usability of instructional augmented reality apps. By arranging data into different

layers of detail, the interface can be laid out with reduced risk of cluttering and with natural hints that pull the user toward the targets required for tracking. Another tool to help the user learn how to use AR is the target outlines that show how to aim the camera for tracking.

The app's non-AR GUI is used as a bridge between the user and the AR, intended to help the user with the process of triggering. Both the index and the GUI guides tell the user where to look. When using a GUI guide, the app will know which target the user should be aiming the camera at and display its outline on the screen if necessary.

The ability to freeze the camera at any point enables the user to follow guides and read information without continuing to hold the device in the same position and orientation. This allows the user to put the device down or hand it to a friend without losing the AR interface.

Tracking is without doubt the single largest usability issue. We have studied and described ways to improve it and ways to work around it. Extended tracking helps tremendously, as it lets the augmented reality interface remain in a good position even when normal tracking fails. Unfortunately there are factors developers cannot currently do much about, such as the requirement for good lighting and highly detailed, contrast-rich targets.

8.3 Future Work

8.3.1 Other Tracking Techniques

As we currently use Vuforia's 2D image processing feature to recognize the target in our application an interesting continuation for this thesis would be to research the possibilities of using 3D tracking techniques such as tracking of arbitrarily shaped 3D objects. One could also investigate using techniques like *SLAM* (simultaneous localization and mapping). This could potentially increase the application's tracking quality and give more room for development of better and more precise instructions and guides.

8.3.2 Continuation of Generalization

As explained in section 6.3 we have laid the groundwork for having the application work with multiple different machine models. However not

much testing has been done partly since we only had one model to test on but also because we did not prioritize it during development. If we were to continue to develop the prototype, more time could be spent on developing an effective way to build AR applications for different models of the same type of machine.

8.3.3 Development for Wearable Devices

It is undeniable that instructional AR applications would work best for hardware devices that are not required to be held by the user, like AR glasses. Wearable AR devices present many new challenges, partly due to their smaller size. For instance, it is possible that this kind of hardware requires different interaction techniques but that the animated guides and instructions can remain similar to the mobile application's. An interesting continuation would be to implement a similar prototype for AR glasses and compare the differences. This would lead to a better understanding of how big of an improvement it is to use glasses instead of a mobile device. Although mobile devices have an edge in that many potential customers already own smartphones.

9 References

[1] Ronald Azuma and Chris Furmanski. Evaluating label placement for augmented reality view management. In *Proceedings of the 2nd IEEE/ACM international Symposium on Mixed and Augmented Reality*, page 66. IEEE Computer Society, 2003.

[2] Ronald Azuma, Jong Weon Lee, Bolan Jiang, Jun Park, Suya You, and Ulrich Neumann. Tracking in unprepared environments for augmented reality systems. *Computers & Graphics*, 23(6):787–793, 1999.

[3] Ronald T Azuma. A survey of augmented reality. *Presence*, 6 (4):355–385, 1997.

[4] Huidong Bai, Gun A Lee, and Mark Billinghurst. Freeze view touch and finger gesture based interaction methods for handheld augmented reality interfaces. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, pages 126–131. ACM, 2012.

[5] Julie Carmigniani and Borko Furht. Augmented reality: an overview. In *Handbook of augmented reality*, pages 3–46. Springer, 2011.

[6] Daqri. Anatomy 4D, 2014. URL `http://site.daqri.com/project/anatomy-4d/`. [Online; accessed 7-May-2014].

[7] Elizabeth FitzGerald, Anne Adams, Rebecca Ferguson, Mark Gaved, Yishay Mor, and Rhodri Thomas. Augmented reality and mobile learning: the state of the art. In *CEUR Workshop Proceedings*, volume 955, pages 62–69, 2012.

[8] Joseph L Gabbard and J Edward Swan. Usability engineering for augmented reality: Employing user-based studies to inform design. *Visualization and Computer Graphics, IEEE Transactions on*, 14(3):513–525, 2008.

[9] Michael Gervautz and Dieter Schmalstieg. Anywhere interfaces using handheld augmented reality. *IEEE Computer*, 45(7): 26–31, 2012.

[10] Metaio GmbH. Tips to create image trackable, 2014. URL `http://dev.metaio.com/sdk/tracking-config/create-image-trackable/`. [Online; accessed 20-March-2014].

[11] Google. Use Google camera, 2014. URL `https://support.google.com/nexus/answer/`

2840311?hl=en. [Online; accessed 20-May-2014].

[12] Michael Haller, Mark Billinghurst, and Bruce H Thomas. *Emerging technologies of augmented reality: interfaces and design*. Igi Global, 2007.

[13] Jan Herling and Wolfgang Broll. Markerless tracking for augmented reality. In *Handbook of Augmented Reality*, pages 255–272. Springer, 2011.

[14] IN2AR. Image marker guide, 2014. URL http://www.augmented-reality-games.com/markerguide.asp. [Online; accessed 20-March-2014].

[15] Qualcomm Connected Experiences Inc. Extended tracking, 2014. URL https://developer.vuforia.com/resources/dev-guide/extended-tracking. [Online; accessed 21-May-2014].

[16] Qualcomm Connected Experiences Inc. Image target enhancement tricks, 2014. URL https://developer.vuforia.com/resources/dev-guide/image-target-enhancement-tricks. [Online; accessed 20-March-2014].

[17] Qualcomm Connected Experiences Inc. Natural features and rating, 2014. URL https://developer.vuforia.com/resources/dev-guide/natural-features-and-rating. [Online; accessed 20-March-2014].

[18] Tai-Wei Kan, Chin-Hung Teng, and Mike Y Chen. Qr code based augmented reality applications. In *Handbook of Augmented Reality*, pages 339–354. Springer, 2011.

[19] Christian Koch, Matthias Neges, Markus König, and Michael Abramovici. Performance study on natural marker detection for augmented reality supported facility maintenance. In *Australasian Journal of Construction Economics and Building-Conference Series*, volume 2, pages 23–34, 2014.

[20] Audi Metaio. Audi ekurzinfo app, 2013. URL http://www.metaio.com/press/press-release/2013/metaio-audi-ag-release-interactive-augmented-reality-manual/. [Online; accessed 16-May-2014].

[21] Audi Metaio. Audi ekurzinfo app itunes description, 2014. URL https://itunes.apple.com/gb/app/audi-ekurzinfo/id436341817. [Online; accessed 16-May-2014].

[22] Lars Müller, Ilhan Aslan, and Lucas Krüßen. Guideme: A mobile augmented reality system to display user manuals for home appliances. In *Advances in Computer Entertainment*, pages 152–167. Springer, 2013.

[23] Ulrich Neumann and Jun Park. Extendible object-centric tracking for augmented reality. In *Virtual Reality Annual International Symposium, 1998. Pro-*

ceedings., IEEE 1998, pages 148–155. IEEE, 1998.

[24] Karl Scotland. Aspects of kanban, 2014. URL `http://www.methodsandtools.com/archive/archive.php?id=104`. [Online; accessed 09-May-2014].

[25] Vuforia. Virtual buttons explenation, 2014. URL `https://developer.vuforia.com/resources/sample-apps/virtual-button-sample-app`. [Online; accessed 16-May-2014].

[26] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202. IEEE Computer Society, 2008.

På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Filip Källström, Fredrik Palm